VBASICA for the PlusPC

You can run VBASICA on your PlusPC either in I mode or in V mode. VBASICA is similar in each mode except that several enhanced features are implemented in V mode only. Thus, you can write a program for one mode and run the program in the other mode with few changes to the program.

The information contained in the VBASICA manual is for both modes. This beginning section describes the differences between using VBASICA in I mode and using VBASICA in V mode. It also describes the enhanced VBASICA features that you can use only in V mode.

At the end of this section is a list of some VBASICA features that are documented in the manual but are not implemented in this version of VBASICA.

This section contains the following information:

- Running VBASICA in V and I modes
- The enhanced features for V mode:
  - —the LCOPY statement
  - —the VBCONF program
  - —graphics screen types and modes
- Features not implemented in this version of VBASICA for the PlusPC

Running VBASICA in I Mode and in V Mode

On the PlusPC, a loading program loads VBASICA. If you are in I mode, the loading program loads the VBASICA designed for I mode. If you are in V mode, the loading program loads the VBASICA designed for V mode. To run the loader, type:

BASICA(cr)

Note: An MS-BASIC binary file must be saved as an ASCII file before you can use it under VBASICA on the PlusPC. After you save the ASCII file, you can load it into VBASICA. Because an ASCII file loads more slowly, you can save the file from VBASICA as a binary file.

Enhanced Features for V Mode

This section describes the enhanced features that you can use in V mode: the LCOPY statement, the VBCONF program, and graphics screens and modes.

LCOPY Statement

FORMAT:

LCOPY

PURPOSE:

Dumps the screen display to a graphics printer.

REMARKS:

With the LCOPY statement, VBASICA can output both text and graphics to a dot matrix printer. VBASICA supports seven printers; you can also define your own. Before you can use LCOPY, VBASICA must be installed for the printer you are using. Thus, a VBASICA program—VBCONF.BAS—is included on the distribution diskette to install VBASICA for you. VBCONF is described in the next section.

Before running this program, you should back up your master diskette and work from a copy.

After VBASICA is configured, LCOPY can use only the printer specified in the sign-on message.

EXAMPLE:

```
10 SCREEN 2     'set up for some graphics
20 KEY OFF      'turn off the function key display
30 CLS 2        'clear the screen
40 FOR RADIUS = 20 TO 200 STEP 10
50 CIRCLE (400,200),RADIUS     'draw some graphics
60 NEXT RADIUS
70 LCOPY        'copy the screen to the printer
80 END
```

The VBCONF Program

To run the VBCONF program from the operating system, enter the following command in response to the operating system prompt:

BASICA VBCONF(cr)

Or from VBASICA, enter the following command from the operating system prompt:

RUN "VBCONF"(cr)

The program displays the following menu choices:

```
ABORT - Abort this program without changing VBASICA.
VICTOR Printer 6010/6020
VICTOR Printer 6015/6025
VICTOR Printer 6070/6075
Tally Printer (160s/180s)
Tally Printer (140)
C. Itoh Printer (8510A, 8510S/1550A, 1550S)
Epson MX-80/MX-100 (with GRAFTRAX)
Epson FX-80/FX-100
Okidata Printer (U84 Only)
No Printer - disable LCOPY
```

Use the cursor arrow keys to move the reverse video bar up and down over the menu. When the choice you want is highlighted, press Return. The V9BASICA.XEC and VIBASICA.XEC files on the default drive are modified and you return to the operating system. If you choose to ABORT, you return to VBASICA. If you choose "No Printer", LCOPY returns an "Illegal function call" error when you try to use it.

The VBASICA sign-on message displays the name of the installed printer. Rerun VBCONF any time you change printers.

You can install a printer not listed in the menu by modifying the VBCONF.BAS source code. After reading your printer's manual and the comments in the program, you should be able to configure VBASICA for most printers that support bit-mapped dot matrix printing.

VICTOR dot matrix printers have special programs built into the hardware ("firmware") that make the printers run faster. These programs also correct the aspect ratio between the printer and the screen so that circles are displayed as round and not slightly oval.

Graphics Screens and Modes in V Mode

VBASICA can create medium- or high-resolution graphics on the color screen or on the standard screen. If your computer has a color card you can put color and black-and-white images on an attached color screen. VBASICA in V mode supports three screen types:

- Your standard screen
- A software-simulated IBM color screen displayed on your standard screen
- The color screen, which requires a color screen and the VICTOR IBM-compatible color card

From VBASICA, you can use the SCREEN statement to change the resolution mode. There are three modes:

- Mode 0: A text-only video display mode
- Mode 1: A medium-resolution mode for graphics and text
- Mode 2: A high-resolution mode for graphics and text

You can access each screen mode and type with the SCREEN command. As the operating system loads VBASICA, it checks if a color card is installed and if the bit map is allocated. If a color card is installed, VBASICA appears on the color screen. If not, VBASICA comes up on the standard screen. If the bit map is not allocated, the monochrome graphics screens cannot be accessed. (See screens 42, 61, and 62 in Table 1.) To change between screen types, use a type/mode number. Table 1 lists all possible type and mode combinations.

Table 1: Screen Type and Mode Combinations

SCREEN	ROWS	COLS	GRAPHICS PIXELS	COLOR, B/W, COMMENTS
Color				
20	25	40/80	No graphics	Full 16 colors, 8 or 4 pages, depending on width
21	25	40	320 × 200	4 colors, medium resolution
22	25	80	640 × 200	2 colors: black and white
PlusPC				
40	25	40/80	No graphics	PlusPC text mode
41	(Not available; use screen 61)			
42	25	80	400 × 800	PlusPC high-resolution screen
IBM Monochrome				
60	25	40/80	No graphics	IBM text mode
61	25	40	320 × 200	4 shades, medium resolution
62	25	40/80	640 × 200	Simulated IBM high-resolution monochrome mode

After you select a screen type, the mode numbers (0, 1, and 2) work as documented in the SCREEN statement description. Thus, the following command switches to the PlusPC text mode:

```
SCREEN 40
```

If this command is followed by the command:

```
SCREEN 2
```

the PlusPC high-resolution screen is selected. Similarly, the following command loads the color high-resolution screen:

```
SCREEN 22
```

To switch to the color medium-resolution screen, enter the following command:

```
SCREEN 1
```

When you want to change screens you can specify the full mode and type number or just a mode change within the current type.

If you run programs developed under VBASICA and have a color screen, issue the command **SCREEN 20** before you start to run the color and black-and-white programs. If you do not have a color screen, enter the command **SCREEN 60** to simulate the IBM color on the standard screen. Programs written specifically for the PlusPC can take advantage of its higher resolution if you use the command **SCREEN 42**.

The Color Attributes in V Mode

You can specify a color attribute with the graphics statements PSET, PRESET, LINE, CIRCLE, PAINT, and DRAW. The range is 0 to 3. These color attribute numbers are distinct from the numbers that refer to actual colors; the latter are used only as parameters in the COLOR statement.

On screen 61, 0 selects black; 1, 2, and 3 select varying shades of white.

In Mode 1 on the color screen (screen 21), 0 selects the background color; 1, 2, and 3 select foreground colors.

In Mode 2 (screen 22), 0 or 2 selects black; 1 or 3 selects white.

NOTE: The COLOR statement does not affect any graphics screen except screen 21.

Coordinates

The drawing statements PSET, PRESET, LINE, CIRCLE, GET, PUT, and PAINT require you to specify screen locations as pairs of (x,y) coordinates. The format is (<x>,<y>), where <x> and <y> are numeric expressions.

The screen coordinates are shown in Table 2.

Table 2: Screen Coordinates

TYPE/MODE	x (HORIZONTAL)	y (VERTICAL)
61, 21	0–319	0–199
42 (standard screen)	0–799	0–399
62, 22 (color screen)	0–639	0–199

Point (0,0) is the upper left corner.

When you clear the screen with either the SCREEN statement or the CLS statement, the graphics cursor is set to the middle of the screen. Table 3 defines the midscreen coordinates.

Table 3: Midscreen Coordinates

SCREEN	COORDINATES
Mode 21, 61	(160,100)
Mode 42	(400,200)
Mode 22, 62	(320,100)

Features Not Implemented for the PlusPC

The following are not implemented in this release of VBASICA:

- The PAINT statement does not support tiling.
- The LINE statement does not support the style attribute.
- The PLAY statement does not support the incrementing and decrementing octaves option.
- User-defined trappable keys are not supported. Only keys 0 through 11 can be trapped.
- The MS-DOS PATH command and the use of pathnames in file specifiers are not supported.

The following statements, commands, and functions are not supported in this release of VBASICA for either mode:

CHDIR command
ENVIRON statement
ENVIRON$ function
ERDEV function
ERDEV$ function
IOCTL statement
IOCTL$ function
MKDIR command
ON PLAY statement
ON TIMER statement
PLAY OFF statement
PLAY ON statement
PLAY STOP statement
PLAY (n) function
PMAP function
RANDOMIZE statement
RMDIR command
SHELL statement
TIMER OFF statement
TIMER ON statement
TIMER STOP statement
VIEW statement
VIEW PRINT statement
WINDOW statement

VBASICA

Rev. b

VICTOR Publications
380 El Pueblo Road
Scotts Valley, California 95066
(408) 438-6680

First VICTOR printing April, 1985.

ISBN 0-88182-137-3 Printed in U.S.A.

Important Software Diskette Information

For your own protection, do not use this product until you have made a backup copy of your software diskette(s). The backup procedure is described in the user's guide for your computer.

Please read the DISKID file on your new software diskette. DISKID contains important information including:

- The part number of the diskette assembly.
- The software library disk number (for internal use only).
- The date of the DISKID file.
- A list of files on the diskette, with version number, date, and description for each one.
- Configuration information (when applicable).
- Notes giving special instructions for using the product.
- Information not contained in the current manual, including updates, any known bugs, additions, and deletions.

To read the DISKID file onscreen, follow these steps:

1. Load the operating system.
2. Remove your system diskette and insert your new software diskette.
3. Enter —

   type diskid(cr)

4. The contents of the DISKID file is displayed on the screen. If the file is large (more than 24 lines), the screen display will scroll. Type CTRL-S to freeze the screen display; type CTRL-S again to continue scrolling.

Preface

VICTOR BASIC Advanced (VBASICA) is an IBM PC-compatible extension to MS-BASIC. This manual includes MS-BASIC commands and the advanced commands.

VBASICA takes advantage of the facilities of the newer 16-bit microprocessors. The extra capabilities provide:

- Advanced graphics
- Sound
- Disk I/O and telecommunications support
- Event trapping

Using This Reference Manual

Chapter 1 gets you started with VBASICA. Chapter 1 also covers device-independent I/O and files; and both initialization and printer configuration are described.

Chapter 2 discusses the character set and explains how to edit your programs with the full screen editor. This editor provides immediate visual feedback and speeds editing functions such as cursor movement, insertion, and deletion. Chapter 2 also describes how VBASICA handles different data types.

Chapter 3 describes all the VBASICA statements, commands, and functions.

Chapter 4 describes the communications option, including Communication I/O, the TTY program, and Com I/O functions.

There are 7 appendixes and an index. Appendix A describes error messages. Appendix B lists key codes. Appendix C is an ASCII character code table. Appendix D includes the math functions. Appendix E lists the scan codes. Appendix F describes how to convert programs to VBASICA. Appendix G discusses VBASICA disk I/O.

Conventions

This manual uses the following conventions:

- The parts of the command line that you type are shown in uppercase. You can type the command in upper- or lowercase. For example:

```
A>COPY WORK.TXT B:PAYROLL.DOC
```

- Parts of the command line that are optional or variable are shown in lowercase.
- A (cr) in the command line indicates that you must press the Return key.
- In the text, names of commands, files, and programs appear in all-uppercase—for example, COPY or PIZZA.YUM.
- In the text examples, the system prompt is assumed to be A>, unless otherwise specified. Your system prompt might have additional characters.

Contents

TABLES

Chapters

1

Getting Started

1.1 Introduction

This chapter describes VBASICA's special features, such as graphics, device-independent I/O, and event trapping.

1.2 Modes of Operation

When VBASICA is initialized, it types the prompt "Ok." "Ok" indicates that VBASICA is at command level—that is, it is ready to accept commands. At this point, you can use VBASICA in either of two modes: the direct mode or the indirect mode.

In the direct mode, line numbers do not precede VBASICA statements and commands. VBASICA executes them as they are entered. Results of arithmetic and logical operations can be displayed immediately and stored for later use, but the instructions are lost after execution. This mode is useful for debugging and for using VBASICA as a calculator for quick computations that do not require a complete program.

Use the indirect mode for entering programs. VBASICA precedes program lines with line numbers and stores them in memory. You execute the program in memory by entering the RUN command.

1.3 Line Format and Line Numbers

Program lines in a VBASICA program have the following format, with square brackets indicating optional information:

nnnn VBASICA statement [:VBASICA statement...](cr)

You can place more than one VBASICA statement on a line, but each statement on a line must be separated from the last by a colon. A VBASICA program line always begins with a line number, ends with a carriage return, and can contain up to 255 characters.

1.4 Initialization

FORMAT:

BASICA [< filename >] [/F: < number of files >] [/S: < lrecl >]
[/C: < buffer size >] [/M: < highest memory location >]

REMARKS:

Load and execute VBASICA by typing the following command at the DOS command line prompt:

BASICA

Upon loading, VBASICA responds with the banner:

```
GW-BASIC 2.01
(1) Copyright Microsoft 1983, 1984
nnnn Bytes Free
Ok
```

You can alter the VBASICA operating environment somewhat by specifying option switches following VBASICA on the command line; these switches are described in the next paragraphs.

< **filename** > The filename of a VBASICA program. Files and filenaming conventions are described in the next section. If < filename > is present, VBASICA proceeds as if you gave a RUN " < filename > " command after initialization. VBASICA assumes a default file extension of .BAS if none is given. If you use this form of the command line in VBASICA programs, you can run the programs in batch files. Such programs must exit with the SYSTEM command so VBASICA can execute the next command in the batch file.

/F: < **number of files** > If present, sets the maximum number of files that can be open simultaneously during the execution of a VBASICA program.

Each file requires 62 bytes for the File Control Block (FCB) plus 128 bytes for the data buffer. The data buffer size can be altered via the /S: option switch. If the /F: option is omitted, the number of files defaults to 3.

/S: < **lrecl** > If present, sets the maximum record size allowed for use with random files. **Note:** The record size option to the OPEN statement cannot exceed this value. If you omit the /S: option, the record size defaults to 128 bytes.

/C: < **buffer size** > If present, controls RS-232-C communications. If /C:0, RS-232-C support is disabled. Any subsequent I/O attempts result in the "Device Unavailable" error. Specifying /C: < n > allocates < n > bytes for the receive buffer and 128 bytes for the transmit buffer for each port. If you omit the /C: option, 256 bytes are allocated for the receive buffer and 128 bytes for the transmit buffer of each card present.

/M: < highest memory location > When present, sets the highest memory location that VBASICA uses. VBASICA attempts to allocate 65K of memory for the data and stack segments. To use machine language subroutines with VBASICA programs, use the /M: switch to reserve enough memory for them.

1

NOTE: < number of files >, < lrecl >, < buffer size >, and < highest memory location > can be decimal, octal (preceded by &O), or hexadecimal (preceded by &H).

EXAMPLE:

Use all memory and 3 files; load and execute PAYROLL.BAS:

```
A>BASICA PAYROLL
```

Use all memory and 6 files; load and execute INVENT.BAS:

```
A>BASICA INVENT/F:6
```

Disable RS-232-C support and use only the first 32K of memory:

```
A>BASICA /C:0/M:32768
```

Use 4 files and allow a maximum record length of 512 bytes:

```
A>BASICA /F:4/S:512
```

Use all memory and 3 files; allocate 512 bytes to RS-232-C receive buffers and 128 bytes to transmit buffers, load and execute TTY.BAS:

```
A>BASICA TTY/C:512
```

1.5 Files

A file is a collection of data not stored in the computer's random access memory (RAM). Instead, the data is stored on disk or some other device. Several commands save and retrieve file information, such as SAVE, LOAD, and LIST.

VBASICA supports the concept of device-independent I/O (input/output) files. Consequently, you can treat any type of input/output as I/O to a file—whether you are using a diskette or a printer, or whether you are communicating with a device.

1.5.1 File Specification

The physical file is described by its filename and specification, or filespec. The filespec can include pathnames for VBASICA 2.0 and later versions. A simple filename is a sequence of characters that can optionally be preceded by a drive designation, be devoid of backslashes, and be optionally followed by an extension.

The filespec is a string expression of the form:

[< pathname >][< device >][< filename >]

< pathname > is the sequence of directory names as described in Chapter 1.5.2.

< device > is a physical device:

KYBD:	Keyboard	Input only
SCRN:	Video Display	Output only
LPT1:	First Line Printer	"
LPT2:	Second Line Printer	"
LPT3:	Third Line Printer	"
COM1:	RS-232-C Port A	Input/Output
COM2:	RS-232-C Port B	"
A: to O:	Disk Drives	"

< filename > is the name given to the file. The name conforms to the VBASICA filename conventions and consists of two parts separated by a period:

< filename > [. < extension >]

The filename can have from 1 to 8 characters and the extension from 0 to 3 characters.

VBASICA uses a default extension of .BAS on LOAD, SAVE, MERGE, RUN, CHAIN, BLOAD, and BSAVE commands if no period appears in the filename and if the filename has fewer than 9 characters. If you do not specify the device, the current VBASICA default disk drive is assumed.

File specification for communications devices differs slightly. Replace the filename with a list of options specifying such items as baud rate and parity.

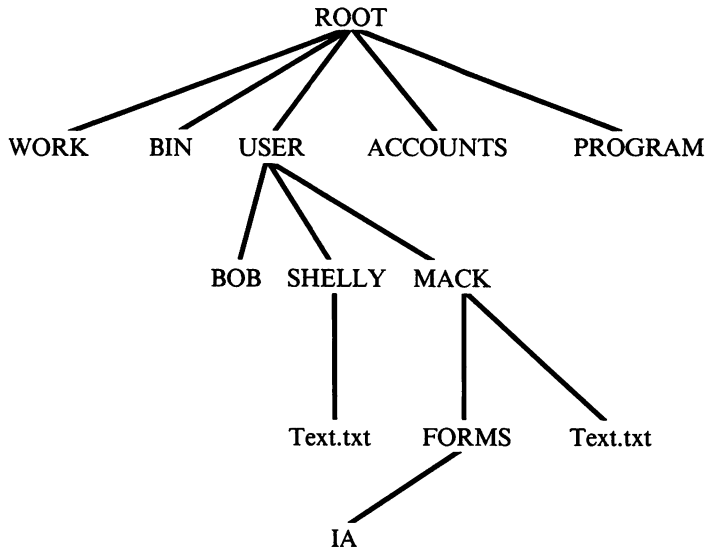
The following statements, commands, and functions support device-independent I/O. For more information, see the descriptions in Chapter 3.

BLOAD	INPUT$	LPOS	PRINT USING
BSAVE	KILL	LPRINT	PUT
CHAIN	LINE	MERGE	RESET
CLOSE	LIST	NAME	RUN
EOF	LLIST	OPEN	SAVE
FILES	LOAD	OPEN COM	WIDTH
GET	LOC	POS	WRITE
INPUT	LOF	PRINT	

1.5.2 Pathnames

A pathname is a sequence of directory names followed by a simple filename, each separated from the previous one by a backslash (\), and no longer than 63 characters. If a device is specified, it must be specified at the beginning of the pathname. The pathname format is:

[< d > :][\] < directory > \] [< directory > \...][< filename >]



Sample Hierarchical Directory Structure

In the structure shown above, directories are in all uppercase letters. The two entries named Text.txt and the entry named IA are files.

If a pathname begins with a backslash, DOS searches for the file beginning at the root (or top) of the tree. Otherwise, DOS begins at the user's current directory, known as the working directory, and searches downward from there.

\USER\SHELLY\TEXT.TXT is the pathname of Shelly's Text.txt file.

When you are in your working directory, a filename and its corresponding pathname may be used interchangeably. The following are some sample names:

\
indicates the root directory.

\PROGRAMS
Sample directory under the root directory containing program files.

\USER\MACK\FORMS\IA
A typical full pathname. This example is a file named IA in the directory named FORMS belonging to a subdirectory of USER named MACK.

USER\SHELLY
A relative pathname; it names the file or directory SHELLY in subdirectory USER of the working directory. If the working directory is the root (\), it names \USER\SHELLY.

Text.txt
Name of a file or directory in the working directory.

DOS provides special shorthand notations for the working directory and the parent directory (one level up) of the working directory:

. (one period)
DOS uses this shorthand notation to indicate the name of the working directory in all hierarchical directory listings. DOS automatically creates this entry when you make a directory.

.. (two periods)
The shorthand name of the working directory's parent directory.

If you type the following command, DOS lists the files in the parent directory of your working directory:

```
DIR ..
```

If you type the next command, DOS lists the files in the parent's PARENT directory:

```
DIR ..?..
```

Working with Pathnames in VBASICA

Not only can VBASICA provide the ability to access files from other directories using pathnames, but you can also use it to create, change, and remove paths, using the VBASICA commands MKDIR, CHDIR, and RMDIR. For example:

- The VBASICA statement MKDIR "ACCOUNTS" creates a new directory, ACCOUNTS, in the working directory of the current drive.
- The VBASICA statement CHDIR "B:EXPENSES" changes the current directory on drive B to EXPENSES.
- The VBASICA statement RMDIR "CLIENTS" deletes an existing directory, CLIENTS, as long as that directory was empty of all files with the exception of "." and "..".

For further information on using paths in VBASICA, see the CHDIR, ENVIRON, ENVIRON$, MKDIR, and RMDIR statements in Chapter 3.

1.6 Re-Direction of Standard Input and Standard Output

VBASICA can be re-directed to read from standard input and write to standard output by providing the input and output filenames on the command line:

BASICA [program name] [< input file] [> output file]

Note that the characters "<" before the input file and ">" before the output file are literally those characters, and not angle brackets indicating a required argument. If two greater-than characters (>>) appear before the output filename, the output is appended to that file.

RULES:

1. When redirected, all INPUT, LINE INPUT, INPUT$, and INKEY$ statements will read from the input file.
2. If the program does not specify a file number in a PRINT statement, that output is redirected to the declared output file instead of the standard output device, the screen.
3. Error messages go to standard output.
4. File input from "KYBD:" still reads from the keyboard.
5. File output to "SCRN:" still outputs to the screen.
6. VBASICA continues to trap keys from the keyboard when you use the ON KEY(n) statement.
7. Pressing CTRL and PrtSc simultaneously does not cause LPT1: echoing if standard output is redirected.
8. Typing CTRL-BREAK causes VBASICA to close any open files, issue the message "Break in line < line number > " to standard output, and exit VBASICA.
9. When input is redirected, VBASICA continues to read from this source until it detects an end-of-file character. This condition may be tested with the EOF function. If the file is not terminated by a CTRL-Z or if a VBASICA input statement tries to read past end-of-file, then any open files are closed, the message "Read past end" is written to standard output, and VBASICA terminates.

EXAMPLES:

```
BASICA MYPROG > DATA.OUT
```

Data read by INPUT and LINE INPUT continues to come from the keyboard. Data output by PRINT goes into the file DATA.OUT.

```
BASICA MYPROG < DATA.IN
```

Data read by INPUT and LINE INPUT comes from DATA.IN. Data output by PRINT continues to go to the screen.

```
BASICA MYPROG < MYINPUT.DAT > MYOUTPUT.DAT
```

Data read by INPUT and LINE INPUT now comes from the file MYINPUT.DAT and data output by PRINT goes into the file MYOUTPUT.DAT.


```
BASICA MYPROG < \SALES\JOHN\TRANS >> \SALES\SALES.DAT
```

Data read by INPUT and LINE INPUT now comes from the file \SALES\JOHN\TRANS. Data output by PRINT is appended to the file \SALES\SALES.DAT.

1.7 Graphics, Screens, and Printers

VBASICA can create high-resolution graphics on the color screen. The standard monochrome screen supports text but it does not support graphics; color will support graphics, text, and color. If your computer has a color board, you can put color and black and white images on an attached color screen. VBASICA can work on either a monochrome or color screen. If you have both monochrome and color, use the DOS MODE command to select either monochrome or color before you start VBASICA.

After you are in VBASICA, you can use the SCREEN statement to change the resolution mode. Three modes exist:

- Mode 0: A text-only video display mode.
- Mode 1: A medium-resolution mode for graphics and text.
- Mode 2: A high-resolution mode for graphics and text.

You can produce a printout (or hard copy) of a screen display in VBASICA. If you are in Graphics mode (mode 1 or 2), GRAPHICS.COM is required; you must load it before entering VBASICA. If you are in Text mode, GRAPHICS.COM is not required. GRAPHICS.COM works only with Epson MX and FX series-compatible printers. To produce a hardcopy of your screen display, press the Shift and PrtSc (Print Screen) keys simultaneously.

The VBASICA statements that draw and manipulate images are the following:

PSET	CIRCLE	PAINT
PRESET	GET	DRAW
LINE	PUT	

You can also use the POINT function in graphics. For more information on each of these commands, see Chapter 3. The SCREEN statement describes how to choose a mode, and the COLOR statement discusses the use of colors.

1.7.1 The Color Attributes

You can specify a color attribute with the graphics statements: PSET, PRESET, LINE, CIRCLE, PAINT, and DRAW. The range is 0 to 3. These color attribute numbers are distinct from the numbers referring to actual colors; the latter are only used as parameters in the COLOR statement.

On screen 1, 0 selects black; 1, 2, and 3 select varying densities of white.

In Mode 1 on the color screen (screen 1), 0 selects the background color; 1, 2, and 3 select foreground colors.

In Mode 2 (screen 2), 0 or 2 selects black; 1 or 3 selects white.

NOTE: The COLOR statement does not affect any graphics screen except screen 1.

1.7.2 Coordinates

The drawing statements PSET, PRESET, LINE, CIRCLE, GET, PUT, and PAINT require screen locations as pairs of (x,y) coordinates. The format is (<x>,<y>) where <x> and <y> are numeric expressions. The screen coordinates are shown in Table 1-1.

1

Table 1-1: Coordinates for Color Screen

MODE	x (HORIZONTAL)	y (VERTICAL)
1	0-319	0-199
2	0-639	0-199

Point (0,0) is the upper left corner.

You can specify any integer coordinate value (from −32768 to 32767) for <x> and <y>. VBASICA 2.0 and later releases clip out-of-range coordinates.

You can specify relative coordinates with the statements PSET, PRESET, LINE, and CIRCLE. In the following example you can write <x offset> and <y offset> as numeric expressions:

```
PSET STEP (1,1)
```

VBASICA adds their values to the current graphics cursor to determine the coordinate. The graphics cursor is the point on the screen where the last graphics point was referenced.

All the graphics statements (excluding the POINT function) update the most recent point used. If VBASICA uses the relative form on the second coordinate, it is relative to the first coordinate. In this case you can use the following:

```
STEP (<x offset>,<y offset>)
```

When you clear the screen with either the SCREEN or CLS statement, the graphics cursor is set to the middle of the screen. Table 1-2 defines the midscreen coordinates.

Table 1-2: Midscreen Coordinates for Color Screen

SCREEN	COORDINATES
Mode 1	(160,100)
Mode 2	(320,100)

1.8 Event Trapping

A program can transfer control to a specific program line when a certain event occurs with event trapping. Control is transferred as if a GOSUB statement was executed to the trap routine starting at the specified line number.

The trap routine executes a RETURN statement after completing the event. The program then resumes execution where it was before the event trap.

1.8.1 Event Specifiers

The following are event specifiers:

COM (n) n is the number of the communications channel, 1 or 2. Typically, the COM trap routine reads an entire message from the COM port before returning. **NOTE**: At high baud rates, the interrupt buffer for COM can overflow. Use the COM routine for single-character messages with discretion.

KEY (n) n is a function key number, 1–4. 1 through 10 are the soft keys 1 through 10. 11 through 14 are the cursor direction keys, as follows: 11–Up, 12–Left, 13–Right, 14–Down. A KEY (0) ON, OFF, or STOP enables, disables, or stops all 14 key events.

Note that KEY (n) ON is not the same statement as KEY ON. KEY ON displays the values of all the function keys on the twenty-fifth line of the screen.

When VBASICA traps a key, that occurrence of the key is destroyed. Therefore, you cannot use the INPUT or INKEY$ statements to find out which key caused the trap. If you want to assign different functions to particular keys, you must set up a different subroutine for each key. You cannot assign the various functions with a single subroutine.

1.8.2 Controlling Event Trapping

VBASICA controls event trapping with the following statements:

< event specifier > ON
< event specifier > OFF
< event specifier > STOP

When an event is ON and you specify a nonzero number for the trap, VBASICA checks if the specified event occurred before it starts each new statement. For example, it checks if a function key was struck or a COM character came in. If the event occurred, VBASICA performs a GOSUB to the line you specify in the ON statement.

When an event is OFF, no trapping occurs and VBASICA does not remember the event even if it occurs.

When you specify STOP, no trapping can occur. But if the event happens, VBASICA remembers it and an immediate trap occurs when VBASICA executes an < event > ON.

When you make a trap for a particular event, the trap automatically causes a stop on that event so recursive traps can never take place. The return from the trap routine automatically turns that event trap back on unless VBASICA performs an explicit OFF inside the trap routine. When an error trap occurs, it automatically disables all trapping. Trapping never takes place when VBASICA is not executing a program.

1.8.3 Additional Controls

Event trapping includes the following statements:

ON < **event specifier** > **GOSUB** < **line number** >

1

This statement sets up an event trap line number for the specified event. A < line number > of 0 disables trapping for this event.

RETURN < **line number** >

This optional form of RETURN is primarily intended for use with event trapping. The event trap routine might want to go back into the VBASICA program at a fixed line number while still eliminating the GOSUB entry that the trap created.

Use this nonlocal RETURN with care. Any other GOSUB, WHILE, or FOR active at the time of the trap remains active. If the trap returns from a subroutine, any attempt to continue loops outside the subroutine results in the "NEXT without FOR" error.

2

Data Entry and Editing in VBASICA

This chapter introduces the VBASICA character set and explains how to edit your programs with the Full Screen Editor. A description of how VBASICA handles different data types follows.

2.1 Character Set

The VBASICA character set consists of alphabetic, numeric, and special characters. The alphabetic characters in VBASICA are the upper- and lowercase letters of the alphabet. The numeric characters are the digits 0 through 9.

2.1.1 Special Characters and Keys

Table 2-1 shows the special characters and keys VBASICA uses.

Table 2-1: Special Characters

CHARACTER	NAME
	Blank
=	Equal sign or assignment symbol
+	Plus sign
−	Minus sign
*	Asterisk or multiplication symbol
/	Slash or division symbol
^	Up-arrow or exponentiation symbol
(	Left parenthesis
)	Right parenthesis
%	Percent
#	Number sign

CHARACTER	NAME
$	Dollar sign
!	Exclamation point
[	Left bracket
]	Right bracket
,	Comma
.	Period or decimal point
"	Quotation mark
'	Single quotation mark (apostrophe)
;	Semicolon
:	Colon
&	Ampersand
?	Question mark
<	Less than
>	Greater than
\	Backslash or integer division symbol
@	At-sign
_	Underscore
Backspace	Deletes last character typed
Escape	Escapes edit mode subcommands
Tab	Moves print position to next tab stop (tab stops are every eight columns)
Return	Terminates a line

2.1.2 Control Characters

VBASICA uses these control characters:

- CTRL-A enters edit mode on the line being typed.
- CTRL-C interrupts program execution and returns to VBASICA command level.
- CTRL-G sounds a tone on your computer.
- CTRL-H is a backspace. It deletes the last character typed.
- CTRL-I is a tab. Tab stops are every eight columns.
- CTRL-J extends the current program line to the next physical line.
- CTRL-R retypes the current line.
- CTRL-S suspends program execution.

- CTRL-Q resumes program execution after a CTRL-S.
- CTRL-U deletes the current line.

2.2 The Full Screen Editor

Using the Full Screen Editor during program development saves you considerable time. You can learn to use the Editor by entering a sample program and practicing the edit commands described in this manual.

2.2.1 Writing Programs

The Full Screen Editor processes any line of text you type while VBASICA is in Direct mode. VBASICA is always in Direct mode after the prompt "Ok" and until you give a RUN command.

Lines of text beginning with a numeric character (digit) are considered program statements. You can extend a logical line over more than one physical line by using the linefeed key (^J). This key opens a blank line on the screen and moves any text after it down one line. A carriage return signals the end of the logical line. When you enter a carriage return the entire logical line is passed to VBASICA.

VBASICA processes program statements in one of four ways:

1. Adds a new line to the program—occurs if the line number is legal (range is 0 through 65529) and at least one nonblank character follows the line number in the line.
2. Modifies an existing line—occurs if the line number matches the line number of an existing line in the program. The text of the newly entered line replaces the existing line.
3. Deletes an existing line—occurs if the line number matches the line number of an existing line and the entered line contains ONLY a line number.

4. Produces an error.

   a. If you attempt to delete a nonexistent line, VBASICA displays the "Undefined line number" error message.

   b. If program memory is exhausted, and you add a line to the program, VBASICA displays the "Out of memory" error message and does not add the line.

You can place more than one VBASICA statement on a line, but each statement on a line must be separated from the last by a colon (:).

A VBASICA program line always begins with a line number, ends with a carriage return, and can contain a maximum of 250 characters.

2.2.2 Editing Programs

Use the LIST command to display an entire program or range of lines on the screen to edit them with the Full Screen Editor. Modify text by moving the cursor with the arrow keys and CTRL-B, CTRL-F, and CTRL-N to the place requiring change. Then perform one of the following functions:

1. Overtype characters.
2. Delete characters to the left of the cursor.
3. Delete words or characters to the right of the cursor.
4. Insert characters at the cursor.
5. Add, or append, characters to the end of the current logical line.

Special keys assigned to the various Full Screen Editor functions perform these actions as described in Chapter 2.2.

A program line is not actually modified within the VBASICA program until you enter a Return. It is sometimes easier to move around the screen and make corrections to several lines. Return to the beginning of each line you changed and press Return. The Return stores the modified lines in the program. As you make changes, the cursor might

be positioned on a line containing a VBASICA message, such as "Ok". When this happens, VBASICA automatically erases the line. The program recognizes its own messages, which are terminated by FF Hex.

It is not necessary to move the cursor to the end of the logical line before pressing Return. The Screen Line Editor remembers where each logical line ends and transfers the whole line, regardless of the cursor position.

2.2.3 Function Keys

The Full Screen Editor recognizes ten special function keys, the arrow keys, the Backspace, the Tab, and the Return. Fourteen control keys exist for moving the cursor on the screen, inserting characters, or deleting words or characters. Table 2-2 lists the keys, their corresponding hexadecimal and decimal codes, and their functions. Table 2-3 describes the functions in detail.

Table 2-2: VBASICA Function Keys

KEY			FUNCTION
01	01	CTRL-A	Edit line buffer
02	02	CTRL-B	Previous word
03	03	CTRL-C	Break (stop program)
05	05	CTRL-E	Erase to end of line
06	06	CTRL-F	Next word
08	08	CTRL-H	Destructive backspace
09	09	CTRL-I	Tab (modulo 8)
0A	10	CTRL-J	Linefeed
0B	11	CTRL-K	Home
0C	12	CTRL-L	Clear screen
0D	13	CTRL-M	Carriage return (enter logical line)
0E	14	CTRL-N	Append to end of line
12	18	CTRL-R	Toggle insert/overtype mode
14	20	CTRL-T	Display next set of function keys; toggles display on/off
15	21	CTRL-U	Clear logical line
17	23	CTRL-W	Delete word
1A	26	CTRL-Z	Clear to end of window
1C	28	→	Cursor right
1D	29	←	Cursor left
1E	30	↑	Cursor up
1F	31	↓	Cursor down
7F	128	DEL	Delete character

Table 2-3: Function Explanations

KEY	DESCRIPTION
CTRL-K HOME	Moves the cursor to the upper left corner of the screen.
CTRL-L CLEAR	Clears the screen and positions the cursor in the upper left corner of the screen.
↑	Moves the cursor up one line.
↓	Moves the cursor down one line.
←	Moves the cursor one column left. When the cursor is advanced beyond the left edge of the screen, it moves to the right side of the screen on the preceding line until it reaches the Home position.
→	Moves the cursor one position right. When the cursor is advanced beyond the right edge of the screen, it moves to the left side of the screen on the next line down until it reaches the end of the screen.
CTRL-F	Moves the cursor to the beginning of the next word. A word is defined as the characters A–Z, a–z, or 0–9, and is delineated by space characters. The next word is defined as the next character to the right of the cursor in the set [A..Z] or [0..9].
CTRL-B	Moves the cursor to the beginning of the previous word.
CTRL-N	Moves the cursor to the end of the logical line. VBASICA appends characters typed from this position to the line.
CTRL-T	Advances function key display on the 25th line to the next set of function keys, and toggles the display on/off.
CTRL-E	Erases to the end of the logical line from the current cursor position. VBASICA erases all physical lines until it finds the terminating carriage return.
CTRL-R	Toggles Insert/Overtype mode. Pressing this key changes the mode to the other mode. VBASICA automatically toggles Insert mode to Overtype mode when you press any cursor movement keys or Return. In Insert mode, VBASICA inserts typed characters at the cursor position and moves all characters on the physical line to the right. Wrap-around is in effect: characters advanced off the right edge of the screen appear from the left edge of the screen on the following line. When out of Insert mode, characters typed replace existing characters on the line.

KEY	DESCRIPTION
TAB	In Insert mode, inserts blanks from the current cursor position to the next tab stop. When out of Insert mode, moves the cursor right 8 spaces at a time until it reaches the end of the screen.
DEL	Deletes one character under the cursor for each depression. Then all characters to the right move one position left to fill in the space. If a logical line extends beyond one physical line, characters on subsequent lines are moved left one position to fill in the previous space, and move the character in the first column of each subsequent line up to the end of the preceding line.
BS	Backspace. Deletes the last character typed or the character to the left of the cursor. Moves all characters to the right of the cursor left one position. Moves subsequent characters and lines within the current logical line up, as with the DEL key.
CTRL-U	Erases the entire logical line.
CTRL-C	Returns to Direct mode, without saving any changes made to the line currently being edited.
CTRL-A	Enters the line buffer at the current cursor position for editing.
CTRL-J	Moves to the next physical line; causes scroll, if necessary.
CTRL-W	Deletes characters up to the next word.
CTRL-Z	Clears the screen to spaces from the cursor position to the end of the screen.

You can extend a logical line over more than one physical line by using the linefeed key sequence (CTRL-J). Typing a linefeed causes subsequent text to start on the next line without entering a carriage return. When a carriage return is finally entered, the entire logical line is passed to VBASICA for storage.

Occasionally, VBASICA can return to Direct mode with the cursor positioned on a line containing a message VBASICA issues, such as "Ok". When this happens the line is automatically erased.

NOTE: If you hit a carriage return at a VBASICA message, the message is sent for processing, and a syntax error results.

2.2.4 Syntax Errors

When VBASICA encounters a syntax error during program execution, VBASICA automatically enters EDIT at the line containing the error. For example,

```
10 A = 2$5              (you meant 10 A = 2^5)
RUN
Syntax Error in 10
OK
10 A = 2$5
```

The Screen Line Editor displays the line in error and puts the cursor under the digit 1. You move the cursor right to the dollar sign ($) and change it to an up-arrow (^). Then press Return. VBASICA then stores the corrected line in the program.

Variables are destroyed whenever you change a program line. If you want to examine the contents of some variable before making the change, press CTRL-C instead of moving the cursor. This action returns you to Direct mode, and preserves the variables.

2.3 Constants

Constants are the actual values VBASICA uses during execution. There are two types of constants: string and numeric.

A string constant is a sequence of up to 255 alphanumeric characters enclosed in double quotation marks. The following are examples of string constants:

```
"HELLO"
"$25,000.00"
"Number of Employees"
```

Numeric constants are positive or negative numbers. Numeric constants in VBASICA cannot contain commas. There are five types of numeric constants:

- Integer constants: Whole numbers between −32768 and 32767. Integer constants do not have decimal points.
- Fixed-point constants: Positive or negative real numbers; that is, numbers that contain decimal points.
- Floating-point constants: Positive or negative numbers represented in exponential form (similar to scientific notation). A floating-point constant consists of an optionally signed integer or fixed-point number (the mantissa) followed by the letter E and an optionally signed integer (the exponent). The allowable range for floating-point constants is 10^{-38} to 10^{38}.

  Here are some examples of floating-point constants:

```
235.9881E-7 = .00002359881
2359E6 = 2359000000
```

  (Double-precision floating-point constants use the letter D instead of E. See Chapter 2.3.1.)

- Hex constants: Hexadecimal numbers with the prefix &H. Some examples are:

```
&H76
&H32F
```

- Octal constants: Octal numbers with the prefix &O or &. Examples of octal constants are:

```
&0347
&1234
```

2.3.1 Single- and Double-Precision Numeric Constants

Numeric constants can be single-precision or double-precision numbers. Single-precision numeric constants are stored with up to seven digits. Double-precision numbers are stored with 16 digits of precision, and printed with up to 16 digits.

A single-precision constant is any numeric constant that has seven or fewer digits, an exponential form using E, or a trailing exclamation point (!). These are examples of single-precision constants:

```
46.8
-1.09E-06
3489.0
22.5!
```

A double-precision constant is any numeric constant that has eight or more digits, an exponential form using D, or a trailing number sign (#). These are double-precision constants:

```
345692811
-1.09432D-06
3489.0#
7654321.1234
```

2.4 Variables

Variables are names that represent values in a VBASICA program. The value of a variable is assigned by the programmer, or as the result of calculations done by a program. Before a variable is assigned a value, it is assumed to have a value of zero.

2.4.1 Variable Names and Declaration Characters

VBASICA variable names can be up to 40 characters long. A variable name can contain letters, numbers, and the decimal point. The first character must be a letter. Special type declaration characters are also allowed.

A variable name cannot be a reserved word, but variable names can contain reserved words. Reserved words include all VBASICA commands, statements, function names, and operator names. If a variable begins with FN, it is assumed to be a call to a user-defined function.

Variables represent either a numeric value or a string. String variable names have a dollar sign ($) as the last character. (For example: A$ = "SALES REPORT".) The dollar sign is a variable type declaration character; it "declares" that the variable represents a string.

Numeric variable names declare integer, single-, or double-precision values. The type declaration characters for these variable names are:

%	Integer variable
!	Single-precision variable
#	Double-precision variable

The default type for a numeric variable name is single-precision.

Here are examples of VBASICA variable names:

PI#	declares a double-precision value
MINIMUM!	declares a single-precision value
LIMIT%	declares an integer value
N$	declares a string value
ABC	represents a single-precision value (the default type)

Variable types can also be declared by including the VBASICA statements DEFINT, DEFSTR, DEFSNG, and DEFDBL in the program. These statements are described in Chapter 3.

2.4.2 Array Variables

An array is a group or table of values referenced by the same variable name. Each element in an array is referenced by an array variable that is subscripted with an integer or an integer expression. An array variable name has as many subscripts as there are dimensions in the array. For example, V(10) references a value in a one-dimension array; T(1,4) references a value in a two-dimension array. The maximum number of dimensions for an array is 285. The maximum number of elements per dimension is 32767.

2.4.3 Space Requirements

Table 2-4 shows the space requirements for variables and arrays.

Table 2-4: Space Requirements (in Bytes)

	INTEGER	SINGLE-PRECISION	DOUBLE-PRECISION
Variable	2	4	8
Array (per element)	2	4	8

Strings need three bytes overhead, plus the present contents of the string.

2.5 Type Conversion

When necessary, VBASICA converts a numeric constant from one type to another. You should note the following rules and examples.

If a numeric constant of one type is set equal to a numeric variable of a different type, the number is stored as the type declared in the variable name. For example,


```
10 A% = 23.42
20 PRINT A%
RUN
23
```

If a string variable is set equal to a numeric value (or vice versa), a "Type mismatch" error occurs.

During expression evaluation, all operands in an arithmetic or relational operation are converted to the same degree of precision (that is, the degree of the most precise operand). The result of an arithmetic operation is returned to this degree of precision.

In this example, the arithmetic is done in double-precision, and the result is returned in D# as a double-precision value:

```
10 D# = 6#/7
20 PRINT D#
RUN
 .857142857142857 1
```

In this example, the arithmetic is done in double-precision, and the result is returned in D (a single-precision variable), rounded, and printed as a single-precision value:

```
10 D = 6#/7
20 PRINT D
RUN
 .8571429
```

Logical operators (see Chapter 2.6.3) convert their operands to integers and return an integer result. Operands must be in the range −32768 to 32767; otherwise, an overflow error occurs.

When a floating-point value is converted to an integer, the fractional portion is rounded. For example:

```
10 LET C% = 55.88
20 PRINT C%
RUN
 56
```


If a double-precision variable is given a single-precision value, only the first seven digits (rounded) of the converted number are valid. Consequently, a single-precision value has only seven digits of accuracy. The absolute value of the difference between the printed double-precision number and the original single-precision value is less than 6.3E−8 times the original single-precision value. For example:

```
10 LET A = 2.04
20 LET B# = A
30 PRINT A; B#
RUN
 2.04 2.039999961853027
```

2.6 Expressions and Operators

An expression can be a string or numeric constant, or a variable; or it can combine constants and variables with operators to produce a single value.

Operators perform mathematical or logical operations on values. VBASICA operators are divided into these categories:

2

- Arithmetic
- Relational
- Logical
- Functional

2.6.1 Arithmetic Operators

The order of precedence of arithmetic operators is shown in Table 2-5.

Table 2-5: Arithmetic Operators

OPERATOR	OPERATION	SAMPLE EXPRESSION
^	Exponentiation	X^Y
−	Negation	−X
*, /	Multiplication, floating-point division	X*Y, X/Y
+, −	Addition, subtraction	X+Y

Use parentheses to change the order in which the operations are done. Operations within parentheses are done first. Inside parentheses, the usual order of operations is maintained. Table 2-6 shows some sample algebraic expressions and their VBASICA equivalents.

Table 2-6: Algebraic Expressions

ALGEBRAIC EXPRESSION	VBASICA EXPRESSION
$X + 2Y$	X + Y*2
$X - \frac{Y}{Z}$	X − Y/Z
X (Y/Z)	X*Y/Z
$\frac{X+Y}{Z}$	(X + Y)/Z
$(X^2)^Y$	(X^2)^Y
X(− Y)	X*(− Y)

NOTE: Two consecutive operators must be separated by parentheses.

Integer Division and Modulus Arithmetic

Two additional operators available in VBASICA are integer division and modulus arithmetic.

Integer division is indicated by the backslash (\). (On most keyboards, you type a backslash by pressing the + key and the CTRL key simultaneously.)

The operands are rounded to integers (in the range − 32768 to 32767) before the division is done. The quotient is truncated to an integer; that is, all decimal places are dropped.

Here are two examples of integer division:

```
10\4 = 2
25.68\6.99 = 3
```

In an expression, integer division is performed after all multiplication and floating-point division is finished.

2

Modulus arithmetic is indicated by the operator MOD. Modulus arithmetic gives an integer result equal to the remainder of an integer division.

This example:

```
10.4 MOD 4 = 2
```

is equivalent to 10/4 = 2 with a remainder of 2, but this example:

```
25.68 MOD 6.99 = 5
```

is equivalent to 26/7 = 3 with a remainder 5.

The precedence of modulus arithmetic is just after integer division.

Overflow and Division by Zero

If VBASICA encounters a division by zero while evaluating an expression, the "Division by zero" error message is displayed. Machine infinity with the sign of the numerator is supplied as the result of the division, and execution continues. If the evaluation of an exponentiation results in zero being raised to a negative power, the "Division by zero" error message is displayed. Positive machine infinity is supplied as the result of the exponentiation, and execution continues.

2.6.2 Relational Operators

Relational operators compare two values. The result of the comparison is either true (− 1) or false (0). This result can be used to make a decision regarding program flow. Table 2-7 shows the VBASICA relational operators.

Table 2-7: VBASICA Relational Operators

OPERATOR	RELATION TESTED	EXPRESSION
=	Equality	X = Y
< >	Inequality	X < > Y
<	Less than	X < Y
>	Greater than	X > Y
< =	Less than or equal to	X < = Y
> =	Greater than or equal to	X > = Y

The equal sign is also used to assign a value to a variable.

When arithmetic and relational operators are combined in one expression, the arithmetic is always done first. For example, this expression is true if the value of X plus Y is less than the value of T − 1 divided by Z:

```
X + Y < (T - 1)/Z
```

2.6.3 Logical Operators

Logical operators test multiple relations, bit manipulation, or Boolean operations. The logical operator returns a single-bit result which is either zero (false) or nonzero (true). In an expression, logical operations are performed after arithmetic and relational operations. The outcome of a logical operation is determined as shown in Table 2-8. The operators are listed in order of precedence.

Table 2-8: Logical Operations

VALUES OF X AND Y	NOT X	NOT Y	X AND Y	X OR Y	X XOR Y	X EQV Y	X IMP Y
X = 0	−1						
X = 1	0						
Y = 0		−1					
Y = 1		0					
X = 1 Y = 1	−2	−2	−1	1	0	−1	−1
X = 1 Y = 0	−2	−1	0	−1	−1	0	0
X = 0 Y = 1	−1	−2	0	−1	−1	0	−1
X = 0 Y = 0	−1	−1	0	0	0	−1	−1

Just as relational operators are used to make decisions about program flow, logical operators can connect two or more relations and return a true or false value to be used in a decision. For example:

```
IF D<200 AND F<4 THEN 80
IF I>10 OR K<0 THEN 50
IF NOT P THEN 100
```


Logical operators convert their operands to 16-bit, signed, two's complement integers in the range −32768 to +32767. (If the operands are not in this range, an error results.) If both operands are 0 or −1, logical operators return 0 or −1. The given operation is performed on these integers in bitwise fashion (that is, each bit of the result is determined by the corresponding bits in the two operands).

You can use logical operators to test bytes for a particular bit pattern. For instance, the AND operator can "mask" all but one of the bits of a status byte at a machine I/O port. The OR operator can "merge" two bytes and create a particular binary value. (The two's complement of any integer is the bit complement plus one; that is, NOT X = −(FX = 1).) For example:

```
63 AND 16=16
```

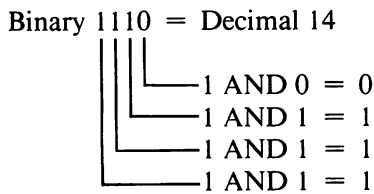
63 can be expressed as binary 111111, and 16 is binary 10000, so 63 AND 16 equals 16.

In this example:

```
15 AND 14=14
```

the answer is computed like this:

Decimal 15 = Binary 1111
Decimal 14 = Binary 1110

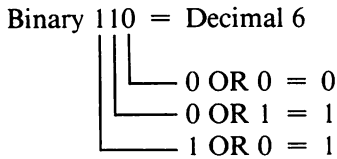


The answer to this expression:

```
4 OR 2=6
```

is computed in this way:

Decimal 4 = Binary 100
Decimal 2 = Binary 010



This expression:

```
10 OR 10=10
```

is computed:

Decimal 10 = Binary 1010
Decimal 10 = Binary 1010

Binary 1010 = Decimal 10

0 OR 0 = 0
1 OR 1 = 1
0 OR 0 = 0
1 OR 1 = 1

Computing logical operations with negative numbers is a little different. Each negative integer must first be converted into its bit complement, and then into its two's complement. Because each word has 16 bits, both of these complements have 16 digit places.

For example, to compute:

```
-1 AND 8=8
```

first convert −1 to its bit complement, binary 1111111111111110. Then add 1 to the bit complement to get the two's complement, 1111111111111111. Then:

Binary 1111111111111111
Decimal 8 = Binary 1000

Binary 1000 = Decimal 8

1 AND 0 = 0
1 AND 0 = 0
1 AND 0 = 0
1 AND 1 = 1

In this example:

```
-1 OR -2=-1
```

the bit complement of −1 is 1111111111111110. Add 1 to get the two's complement, 1111111111111111.

The bit complement of −2 is 1111111111111101, and the two's complement is 1111111111111110. Then:

Binary 1111111111111111
Binary 1111111111111110
1111111111111111 = Decimal −1

2.6.4 Functional Operators

In an expression, a function calls a predetermined operation to be done on an operand. VBASICA has functions that reside in the system, such as SQR (square root) or SIN (sine). These "intrinsic" functions are described in Chapter 3. VBASICA also allows "user-defined" functions.

2.6.5 String Operations

Strings are concatenated using the plus sign (+). For example:

```
10 A$="FILE" : B$="NAME"
20 PRINT A$ + B$
30 PRINT "NEW" + A$ + B$
RUN
FILENAME
NEW FILENAME
```

Strings are compared by using the same relational operators that are used with numbers:

= < > < > < = > =

String comparisons are made by taking one character at a time from each string and comparing their ASCII codes. If all the ASCII codes are identical, the strings are equal. If the ASCII codes differ, the lower code number precedes the higher. If the end of one string is reached, the shorter string is said to be smaller. Leading and trailing blanks are significant. Here are some examples of string comparisons:


```
"AA" < "AB"
"FILENAME" = "FILENAME"
"X&" > "X#"
"CL" > "CL"
"kg" > "KG"
"SMYTH" < "SMYTHE"
B$ < "9/12/85" where B$ = "8/12/85"
```

String comparisons can test string values or alphabetize strings. All string constants used in comparison expressions must be enclosed in quotation marks.

2.7 Input Editing

VBASICA lets you edit your input. Use the EDIT command to change portions of a line without retyping the entire line. EDIT has this format:

EDIT <**line**>

where <line> is the number of the line where editing is to begin.

After you type the EDIT command, VBASICA enters edit mode at the specified line, displays the line number of the line to be edited (followed by a space), and then waits for you to enter an edit mode subcommand.

To delete the entire program in memory, use the NEW command described in Chapter 3. NEW is usually used to clear memory before entering a new program.

2.7.1 Syntax Errors

When VBASICA encounters a syntax error during execution of a program, it automatically enters edit mode at the line that contains the error. For example:

```
10 K = 2(4)
RUN
Syntax error in 10
10
```

When you finish editing the line and press Return (or use the E subcommand), VBASICA reinserts the line. All variable values are lost. To preserve the variable values for examination, exit edit mode with the Q subcommand. VBASICA returns to command level, and all variable values are saved.

2.7.2 CTRL-A

Type a CTRL-A to enter edit mode on the line you are currently typing. VBASICA responds with a carriage return, and the cursor moves to the first character in the line.

If you have just entered a line and want to edit it, the command:

EDIT.

enters edit mode at the current line. (The period refers to the current line.)

2.8 Error Messages

An error message is displayed if VBASICA detects an error that causes program execution to halt.

See Appendix A for a complete list of VBASICA error codes and messages.